REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
08-06-2011	Technical Paper	JUN 2011 - JUL 2011

4. TITLE AND SUBTITLE

On the Robustness of Cognitive Networking Mechanisms to Malicious Insiders

5a. CONTRACT NUMBER
FA8720-05-C-0002

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jonathan Herzog, Gabriel Wachman, and Dan Liu

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02420

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Army
Bldg 2700, Room 3D331
RDER-STI-CM
Fort Monmouth, NJ 07703

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Cognitive networking mechanisms promise to greatly improve network performance over non-cognitive mechanisms, by making more efficient use of bandwidth, spectrum, and power. However, these mechanisms must be designed with cyber security in mind in order to remain efficient in the presence of subverted, adversarial participants. In this paper, we demonstrate the susceptibility of two specific cognitive-networking mechanisms to a single Byzantine participant. Specifically, we describe a novel "energy well" attack against Q-routing, in which a Byzantine participant can attract traffic meant for an honest participant. Secondly, we describe a denial of service attack against a no-regret learning algorithm for Dynamic Spectrum Access (DSA), in which a single Byzantine participant can degrade network-wide performance for an arbitrary amount of time. These attacks demonstrate why cyber security techniques must be designed into cognitive mechanisms before use in the tactical field so that they do not fail to tolerate adversarial behavior. We conclude by discussing possible mitigation concepts and future work.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Zach Sweet
U	U	U	SAR	7	19b. TELEPHONE NUMBER *(include area code)* 781-981-5997

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

MS-55729

On the Robustness of Cognitive Networking Mechanisms to Malicious Insiders

Jonathan Herzog and Gabriel Wachman
Communications Systems and Cyber Security Division
MIT Lincoln Laboratory
Lexington, MA, USA
jherzog@ll.mit.edu, gabriel.wachman@ll.mit.edu

Dan Liu
CERDEC
United States Army
Aberdeen Proving Ground, MD, USA
Daniel.K.Liu@us.army.mil

***Abstract*—Cognitive networking mechanisms promise to greatly improve network performance over non-cognitive mechanisms, by making more efficient use of bandwidth, spectrum, and power. However, these mechanisms must be designed with cyber security in mind in order to remain efficient in the presence of subverted, adversarial participants. In this paper, we demonstrate the susceptibility of two specific cognitive-networking mechanisms to a single Byzantine participant. Specifically, we describe a novel 'energy well' attack against Q-routing, in which a Byzantine participant can attract traffic meant for an honest participant. Secondly, we describe a denial of service attack against a no-regret learning algorithm for Dynamic Spectrum Access (DSA), in which a single Byzantine participant can degrade network-wide performance for an arbitrary amount of time. These attacks demonstrate why cyber security techniques must be designed into cognitive mechanisms before use in the tactical field so that they do not fail to tolerate adversarial behavior. We conclude by discussing possible mitigation concepts and future work.**

I. Introduction

Cognitive networking mechanisms promise to greatly improve network performance (when compared to non-cognitive mechanisms) by making more efficient use of bandwidth, spectrum, power, and so on. They also promise to greatly ease the task of network management by making networks self-aware and self-configuring, to some degree. These advances will benefit both military and civilian networks. Military networks, however, must remain robust and resilient in the face of cyberattacks well beyond what is commonly seen in the civilian sphere. Furthermore, these attacks may be of a very sophisticated nature, or planned well in advance. Such attacks may attempt to subvert network equipment at manufacture or in the supply chain (a *lifecycle* attack), they may attempt to implant malicious/adversarial code on network equipment through so-called 'buffer overflows' (an *injection* attack), and they may attempt to subvert a legitimate operator of network equipment (the *insider* attack).

In all of these cases, a cognitive network may include legitimate participants who are actually acting in an adversarial manner. Although cognitive-networking mechanisms may achieve high and improved performance in non-adversarial settings, this does not guarantee that they will remain robust and high-performing in adversarial ones. As we show in this paper, in fact, cognitive-networking mechanisms may be quite vulnerable to even a single malicious insider. As part of a recent collaboration between MIT Lincoln Laboratory and the U.S. Army Communications-Electronics Research, Development, and Engineering Center (CERDEC) into the security of cognitive-networking mechanisms, we analyzed the security of two specific algorithms:

- The 'Q-routing' algorithm for packet-routing in mobile, ad-hoc networks [1], and
- The 'no regret' algorithm for dynamic spectrum analysis [3].

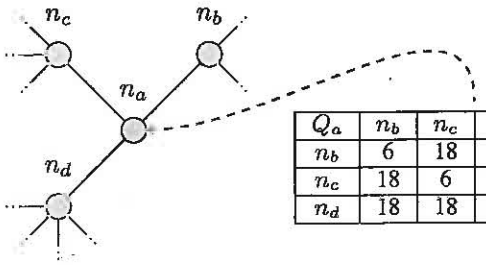
In both cases, we found that the performance of the algorithm can be significantly degraded for a large portion of the network by a single mis-behaving adversary. In this work, we describe these two algorithms and the single-participants attacks we found (Sections II and III). We finish with some observations and conclusions (Section IV).

II. Q-routing

A. Background

At its core, the problem of packet routing in a mobile, ad-hoc network (MANET) routing is a sequence of simple local decisions. At each point in a packet's path from *source* to *destination*, an *intermediate node* comes into possession of the packet and makes a simple-sounding decision: which neighbor should receive the packet next? This can actually be quite difficult to determine in MANETs. First, the network topology of the MANET will change over time—sometimes quite quickly. What was once the 'best' neighbor in the past may not be the 'best' neighbor now. Second, the 'best' neighbor may not be the topologically nearest neighbor. Network congestion may saturate the capacity of links or nodes in a MANET, and so it may sometimes be 'best' to send the packet away from its destination in an effort to reach a less-saturated path. Lastly, the changing topology of a MANET means that the 'best' neighbor of the past may not even *be* a neighbor now.

This work is sponsored by the United States Army under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.



Q_a	n_b	n_c	n_d	n_e	...
n_b	6	18	18	20	...
n_c	18	6	18	30	...
n_d	18	18	6	30	...

Figure 1. Example Q-routing MANET-neighborhood

In this section, we consider one cognitive MANET-routing protocol: Q-routing.

B. The Q-routing algorithm

The Q-routing algorithm [1] is a machine-learning algorithm based on reinforcement learning. As opposed to other MANET-routing protocols (e.g., Optimized Link State Routing (OLSR) [2]) the Q-routing algorithm does not build spanning-trees of the MANET. Instead, each node in a Q-routing MANET requests (and trusts) a time-required-to-deliver estimate from each of its neighbors.

Specifically, every node n_i maintains a table Q_i with one row for each neighbor and one column for each destination in the network. The cell $Q_i(j,d)$, for example, is held in table Q_i (which is maintained by n_i) and associated with

1) n_i's neighbor n_j, and
2) destination d.

In that cell, n_i will maintain a value which estimates how quickly a packet will reach destination d if

- the packet is given to n_i, and
- node n_i passes it on to n_j.

This estimate will be the total of three components:

- The time it takes n_i to process the packet (including the time it stays in n_i's queue),
- The time required for n_i to transmit the packet to n_j, and
- The time it takes for the packet to reach d after being received by n_j.

We illustrate Q-routing with a running example. Consider the MANET-neighborhood of Figure 1. Node n_a maintains the table Q_a, which is shown in part. We note that $Q_a(n_b,n_b)$ is fairly small, as are $Q_a(n_c,n_c)$ and $Q_a(n_d,n_d)$. This makes sense: once a packet for n_b reaches n_a, the only time-consuming events remaining are (1) for the packet to make it through n_a's queue, and (2) for the packet to be transmitted to n_b. The entry in $Q_a(n_c,n_b)$ is larger, as a packet for n_b will take longer to reach n_b if routed through n_c first.

To route a packet for destination d, a node n_i consults its table to find the 'fastest' neighbor. That is, it looks at column d of its table Q_i and finds the row with the smallest entry:

$$\arg\min_n Q_i(n,d)$$

Returning to our example, suppose that n_a receives a packet for destination n_e. It then it consults the column of Q_a for n_e. Looking at that column, it sees that the lowest entry is $Q_a(n_b,n_e)$. Therefore, it forwards the packet to n_b.

The table Q_i makes the 'routing problem' extremely easy to solve—provided that the table is accurate and timely. The core of Q-routing, then, is the mechanism by which the tables are built and updated. The core idea of this process is as follows: when a node n_i routes a packet to a neighbor n_j, it also queries that neighbor about the packet's destination. The neighbor n_j will provide a value from its table, and the original node n_i will use this estimate to update its own table.

Specifically, suppose node n_i routes packets for destination d to its neighbor n_j. Each time it does so, it queries n_j for about its distance to d. Node n_j will look in its table Q_j for the column associated with d, and return the lowest value in that column. Call this value v. When n_i receives a value v from n_j, it understands this to represent the time it will take a packet to arrive at d *once the packet arrives at* n_j. Node n_i wishes, however, to estimate a slightly different value: the time required for a packet to arrive at d once it arrives at n_i, given that it is then passed on to n_j. To estimate this value, it adds to v latency of its queue, s, and the transmission-time to send a packet to n_j, t. Node n_i's new estimate (for time between (a) receiving a packet for d, and (b) the packet arriving at d, given that it routes the packet through n_j) is

$$v+s+t$$

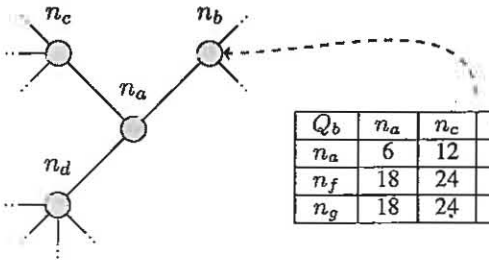
However, node n_i does not immediately change $Q_i(j,d)$ to this value. To do so might make the algorithm overly responsive to transient changes in the network. Instead, node n_i changes $Q_i(j,d)$ to be closer to this new estimate, and does so by applying a discount to the new information.

In particular, let the Q-routing algorithm be parameterized by a *discount factor* $0 \leq \mu \leq 1$. In this case, node n_i adjusts the value in $Q_i(j,d)$ by the following amount:

$$\Delta Q_i(j,d) = \mu\left(v+s+t-Q_i(j,d)\right)$$

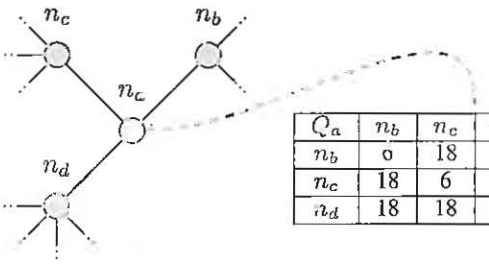
In our running example, suppose that n_a queries n_b about node n_e. Then n_b queries its internal table Q_b about the column associated with n_e. (See Figure 2.) The smallest value in this column is 12, and so n_b returns $d=12$ to n_a. Node n_a adds to this the latency of its own queue ($s=4$) and the transmission time for sending a packet to n_b ($t=2$) and arrives at a new estimate of 18. The value currently in its table for $Q_a(n_b,n_e)$, however, is 20.

Suppose that this instance of Q-routing is running with a discount factor of $\mu=\frac{1}{2}$. Then n_a computes an adjustment



Q_b	n_a	n_c	n_d	n_e	...
n_a	6	12	10	24	...
n_f	18	24	24	24	...
n_g	18	24	24	12	...

Figure 2. Node n_b operation in Q-routing



Q_a	n_b	n_c	n_d	n_e	...
n_b	0	18	18	19	...
n_c	18	6	18	30	...
n_d	18	18	6	30	...

Figure 3. Node n_a's update to table Q_a

of

$$\begin{aligned}\Delta Q_a(n_b, n_e) &= \mu\,(v + s + t - Q_i(j, d)) \\ &= \frac{1}{2}\,(12 + 4 + 1 - 20) \\ &= -1\end{aligned}$$

It then applies this adjustment to $Q_a(n_b, n_e)$ and ends (Figure 3) with a new value of $Q_a(n_b, n_e) = 19$.

C. Our attacks

In our security analysis of the Q-routing algorithm, we uncovered two 'denial-of-service' attacks which could be launched by a single subverted insider:

- The 'random walk' attack: In this attack, the adversary provides false 'distance' values for a particular destination. These values might be random values, chosen afresh each time. Alternately, they can be chosen to slowly increase or decrease at a rate calculated to 'resonate' with the algorithm's discount rate. In both cases, the purpose of the false and challenging values is to prevent the underlying Q-learning algorithm from converging. In Figure 4 (taken from [1]) one can see the performance of the Q-routing algorithm over time, for both light and heavy network loads. (The performance of shortest-path routing is also included for comparison, though we do not need it for this discussion.) As one can see, the Q-routing algorithm displays extremely poor performance at first, while the Q-learning algorithm is still learning the optimal routes. During this phase, packets are being routed more-or-less randomly (hence the name of this attack). When the algorithm converges, on the other hand, performance increases dramatically. The point of this attack, therefore, is to keep the algorithm from converging as long as possible (or to return the algorithm to a pre-convergence state). By changing its reported 'distance' values, either randomly or according to some algorithm, the adversary may be able to accomplish this within some sphere of influence (determined by the discount rate).
- The 'energy well' attack: in this attack, the adversary simply advertises a falsely-small distance to a far-distance destination. This false information eventually propagates to the adversary's neighbors, the neighbors' neighbors, and so on. Eventually, nodes closer to the adversary than the destination will come to believe that the best route to the destination is toward the adversary, which could actually be in the opposite direction from the actual destination.

We note that this second attack may have long-lasting repercussions. Once the adversary has been able to establish the 'energy well,' it has been encoded in the internal table of each fooled node. Even if the adversary is identified and removed from the network, the energy well will continue to exist. In fact, the energy well will continue to trap packets and distort network traffic until it is removed from the internal table of each affected node. Because each node updates its table based on reports from neighbors, and because these neighbors might also have been fooled by the adversary, the energy well will persist for some time. (Ultimately, it is the neighbors of the adversary that will begin to correct their internal tables first, and the effect will spread from the center of the well outward.)

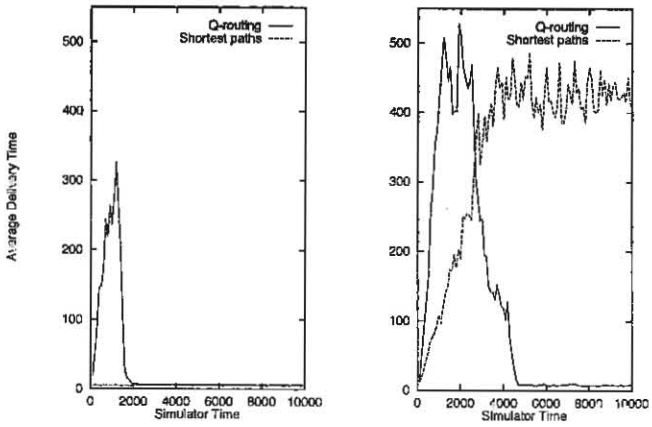


Figure 4. Performance of Q-routing over time. Left chart represents light network load, and the right chart represents heavy network load. Height of curve represents packet-delivery time. (Lower values represent better network performance.) Diagram taken from [1], which included shortest-path routing as a basis for comparison.

D. Our recommendations

Although we do not focus on recommendations or mitigations in this particular work, we note that these attacks can be prevented through network-wide distribution (by each participant) of local information. For example, participants in the Optimized Link-State Protocol (OLSP [2]) broadcast (to the entire network) information about their immediate neighborhood.[1] This information allows every node to construct a spanning-tree of the entire network. In the context of a Q-routing network, this additional information would allow participants to detect (and ameliorate) the above attacks.

III. No-regret learning

A. Background

The term 'dynamic spectrum access' (DSA) refers to the notion that instead of being limited to a single frequency, a radio transmitter might wish to opportunistically 'jump' from frequency to frequency so as to maximize their ability to communicate. In this section, we consider the setting in which many 'peer' transmitters must share a range of frequencies (broken into channels) over which they share equal claim.[2] These users can use any of the available channels, or even multiple channels at once. These channels have finite capacity, however, and can be over-saturated if total usage exceeds their capacity. Furthermore, channel-switching may be an expensive operation which the users wish to avoid as much as possible.

To maximize their efficiency, therefore, each user may wish to observe and predict the behavior of the other users. If it can predict the future behavior of these other agents based on past behavior, then it may be able to identify channels with a high probability of being 'free' for long intervals. It can then use that channel (or a large fraction thereof) for its own communication while minimizing the costs of channel-switching.

B. The algorithm

In this section, we consider one algorithm for channel-selection in this DSA setting. This algorithm [5] draws on an underlying result from *game theory*, showing that non-communicating players can achieve a correlated equilibrium in a repeated game [4]. To apply this result to dynamic spectrum access, setting is modeled as a game wherein the participants are the players and each strategy represents one possible usage of channels. That is, a strategy would have the player to use channel 1 with rate r_1, channel 2 with rate r_2, and so on, for some r_1, r_2,...It is assumed that rates r_i are discretized, and so there are only finitely many rates from which a player might choose. It is further assumed that there are a finite number of channels, leading to a finite number of strategies.

The 'reward' received by each player in the game is proportional to their total throughput over all channels, which is the sum of their throughput on each channel. Their throughput on a given channel, furthermore, is proportional to the rate they choose for that channel—unless the sum of rates on that channel exceeds its capacity. In this case, the throughput of the channel drops to zero, and the players receive no utility from that channel in the outcome of that round.[3] It is in each player's interest, therefore, to use as much as possible of each available channel so long as the channels do not become over-subscribed.

Lastly, we note that the game is a *repeated* game, meaning that secondary users play the above game in each round in an infinite series of rounds. However, players can adapt their strategies between rounds, and can do so based on previous rounds.

Once the DSA setting is represented as a game, the algorithm of this section (presented by [5], which simply refines the more-general algorithm of [4] for the DSA setting) attempts to bring this game to an *equilibrium*: an assignment of strategies (rather, probability-distributions over all possible strategies) to players with the following property: no player can independently (unilaterally) act to increase the overall utility of all players, so long as all other players obey their assigned distribution. Specifically, the algorithm attempts to find a *correlated* equilibrium, which is a type of equilibrium in which the players are given possibly-correlated signals as to the strategies to select. In particular, each player is given their own individual signal, but these signals may be correlated with each other. If the signals are completely uncorrelated, the resulting equilibrium will be a Nash equilibrium. If the signals are correlated, however, the resulting equilibria may produce better overall performance than can be produced by Nash equilibria. In our setting, the semi-common signal is simply the history of previous rounds, and this is enough for the algorithm of [4] to achieve correlated equilibrium—given that the game is unchanging and number of rounds is sufficiently long.

To achieve correlated equilibrium a completely distributed manner, each participant performs a sequence of computations at each round. Specifically, it will compute the *regret* it currently associates with each strategy. The regret of a strategy is defined to be the difference between two values:

- The total utility the node would have 'received' from the game over all previous rounds had it followed its *current strategy* each previous round, and
- The total utility the node would have 'received' from

[1]We note that the authors of the Q-routing algorithm also consider describe a variant algorithm which incorporates this sort of neighborhood-wide communication [1, Section 3.2]. However, their discussion of this variant is motivated by performance concerns rather than security concerns.

[2]The term 'dynamic spectrum access' can also be used in a setting in which some frequencies are licensed to 'primary' users who use them only infrequently. Therefore, they are often available to be used opportunistically by 'secondary' users. We do not consider this setting in this paper.

[3]The model also allows for the possibility that some nodes are out of range from each other, in which case their actions are independent.

the game over all previous rounds had it followed *the strategy in question* each previous round.

The first value is subtracted from the second, and negative values are rounded up to zero. In both cases, the past actions of all other players remain unchanged.

To illustrate this with an example: suppose that a game has just finished its third round, and that participant P is currently using strategy s_0. To calculate the regret it associates with another strategy s_1, it first computes two values:

- First, the utility it would have received had it followed strategy s_0 for all previous rounds (which may or may not have been the case). For the sake of completeness, let us suppose that it would have received utility 1 in round 1, utility 3 in round 2, and utility 3 in round 3.[4] Strategy s_0 then would have generated total utility 7.
- Then, the participant computes the total utility that would have been generated by strategy s_1. With this strategy, the participant would have generated utility 1 in round 1, utility 1 in round 2, and utility 2 in round 3. Thus, strategy s_1 would have generated total utility 4.

With these two numbers, the participant calculates its regret for strategy s_1. Because strategy s_1 would have generated less utility (4) than with its current strategy (which generated utility 7), strategy s_1 is assigned a regret of 0.

Suppose, on the other hand, that the participant also evaluates the regret assigned to another strategy s_2. For the purpose of this example, suppose that this strategy would have generated utility 3 in round 1, utility 2 in round 2, and utility 4 in round 3. Its total utility is 9, which is larger than the utility generated by the current strategy. The regret assigned to s_2 is the difference between the two utility-totals: $9 - 7 = 2$.

If the node associates a positive regret with a strategy, than the node would have been 'better off' adopting that strategy from the beginning. To choose its next strategy, then, the node samples from the following distribution: Let s be the strategy used in the previous round. Then:

- A strategy $s' \neq s$ is assigned a probability proportional to its associated regret. (The proportionality constant is chosen at the beginning of the game in such a way so as to ensure that the sum of these probabilities is less than one.)
- The strategy s is assigned all 'left over' probability mass.

To return to our example, recall that the current strategy of participant P is s_0, and that strategies s_1 and s_2 have regret 0 and 2 assigned to them, respectively. Let us further assume that strategies s_3, s_4 and s_5 have been assigned regrets 3, 4 and 5, respectively, and that these are all possible strategies.

To assign a probability to each strategy, the participant retrieves its proportionality constant. This constant, c, was chosen by the participant at the beginning of the game in such a way so as to be larger than any possible sum of regrets. With this constant, the participant assigns probabilities in the following way:

- s_1: probability 0.
- s_2: probability $2/c$.
- s_3: probability $3/c$.
- s_4: probability $4/c$.
- s_5: probability $5/c$.
- s_0: all remaining probability mass $(1 - \frac{2+3+4+5}{c})$.

It then chooses a strategy for next round according to this distribution.

We note that because the constant c of this example is chosen to be large (so as to ensure that the probability assigned to s_0 will be positive) assigned to strategies s_2 through s_5 may be quite small. Despite this, it is shown (in [4]) that this procedure will 'almost surely' converge to correlated equilibrium for any game. The specific application of this algorithm to DSA channel selection is demonstrated in [5], along with simulation results demonstrating improvements of 5% to 15% over other game-theoretic results.

C. Our attack

As mentioned above, this algorithm is vulnerable to a simple 'denial of service' attack which can be launched by a single subverted participant. We motivate this attack through a simple example:

Suppose that a system of participants has reached convergence, and was been in convergence for some time t. (For the sake of simplicity, suppose that the system has been in convergence for its entire history.) Further suppose that during this time, participant P has used the simple strategy s of using only channel c. By doing so, it has received utility u per round, and therefore received cumulative utility ut.

Now, suppose at time t that the adversary 'floods' channel c. This means that participant P will receive no utility by using channel c. It should switch to a new channel, therefore, and will. But when?

Before the participant will switch, it needs to establish that it 'regrets' not using some other strategy s'. To compute its regret for s', it computes two values:

- The cumulative utility it has received so far, and
- The cumulative utility it would have received by purely following s' from the beginning of history (assuming that all other participants remain as they were).

For the participant to compute positive regret for s', it must be the case that the second value outweighs the first. But when will this be the case?

Let us make the simplifying assumption that strategy s' will yield the participant utility u' (where $u' < u$) per round.

[4] Why are these values not all identical? Because the other participants shifted their strategies from round to round, thus changing the utility of strategy s_0.

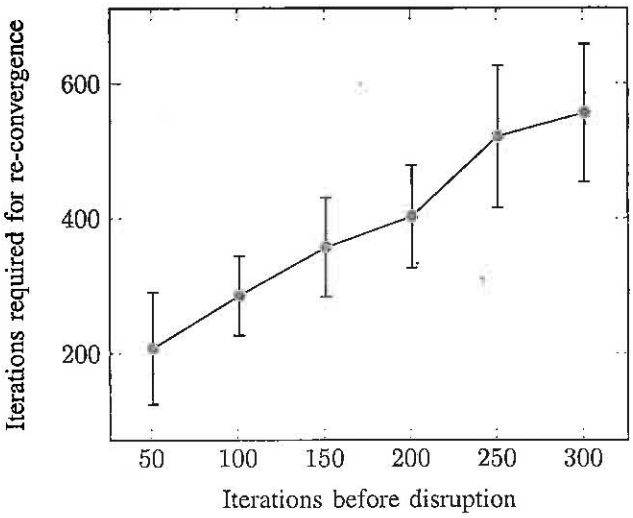


Figure 5. Iterations required for re-convergence, as function of iterations previously in convergence

In this case, let us compute the two values above as they would be computed by the participant at some time $t' > t$:

- Cumulative utility actually received: the participant will have received utility ut until time t, and no utility after that. Thus, a total utility of ut.
- Cumulative utility from strategy s': utility u' for every round, making a total cumulative utility of $u't'$.

Thus, the participant will only 'regret' not using strategy s' when

$$u't' \geq ut$$

Thus, the participant will only switch when $t' \geq \frac{u}{u'}t$. If, for example, $u' = \frac{2}{3}u$, then the participant will only switch when $t' = \frac{3}{2}t$. If $u' = \frac{u}{2}$, then the participant will only switch when $t' = 2t$. In general, the larger the difference between u and u', the longer it will take the participant to switch from strategy s to strategy s'.

This example demonstrates a more general phenomenon: the longer the system has been in convergence, the longer participants will wait to react to disruption. We also confirmed this notion through simulation. By simulating a simple system (three users, two channels), we have been able to demonstrate this intuition experimentally. Figure 5 displays the results. In this experiment, the original system was allowed to come to convergence and to stay in convergence for some number of rounds n. At round n, one of the participants (the adversary) unilaterally increases their use of the channels. When this happens, the two remaining participants will begin to receive sub-optimal utility. They will need to re-allocate the remaining bandwidth between themselves so as to maximize their payouts, which will cause the system to leave convergence. Our simulation allows the system to continue until the two honest participants re-converge to their new, optimal strategies.

Let n' be the number of rounds required to re-converge (which is measured from round n, not round 0). Figure 5 displays n' as a function of n. As can be seen, the longer that a participant has been relieving optimal utility from a convergent system (n), the longer it is willing to tolerate sub-optimal utility before it will switch strategies (n').

Multiple attacks can leverage this characteristic of the regret-based learning system. First, the adversary can apply the simple strategy of our simulation: simply flooding the channels used by a participant of interest. It might also use the 'window of opportunity' to iterate through all possible strategies to find the most disruptive one. Lastly, it can attempt to prevent the system from converging or re-converging by changing its strategies periodically.

D. Our recommendations

The above attacks are possible because of how algorithm-participants regard the past. In particular, when participants decide how much 'regret' to associate with a particular strategy, they compute the total utility they would have received with that strategy over their *entire* history. Specifically, the utility they would have received in the first round is considered to be on par with the utility they would have received in the immediately-previous round, no matter how long ago the first round had been. Put another way, the participants of this protocol do not discount the past. To mitigate this attack, we recommend that the definition of 'regret' be modified to value the immediate past more highly than the distant past. This can be accomplished by simply applying a discount factor which increases over time, or to simply consider a sliding window which encompasses only the most immediate past.

IV. Conclusions

In this work, we considered two cognitive-networking mechanisms—Q-routing for packet-routing in MANETS, and a game-theoretic algorithm for DSA—and demonstrated how both could be subverted by a single malicious insider. This is not to say that both algorithms are inherently vulnerable; both algorithms could be modified to be resistent to the described attacks. (Q-routing could be augmented with some sort of network-wide information-distribution in the style of OLSR, and the DSA algorithm could be modified to react more quickly to disruption.) Nor should this work be interpreted as saying that all cognitive-network algorithms are vulnerable to insider-attacks. However, both attacks described in this paper are enabled by a key aspect of many cognitive-network mechanisms: that network-participants trust other network participants to be acting honestly. By doing so, these cognitive-network mechanisms can more quickly react to new information, or can find optimal configurations which cannot be reached through purely-

local decisions. However, these attacks do demonstrate that cognitive-network mechanisms must be analyzed from a security standpoint before being implemented and deployed on military networks.

REFERENCES

[1] Justin A. Boyan and Michael L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 671–678. Morgan Kaufmann, 1993.

[2] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). Request for Comments 3626, October 2003.

[3] Zhu Han, Charles Pandana, and K. J. Ray Liu. Distributive opportunistic spectrum access for cognitive radio using correlated equilibrium and no-regret learning. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC)*, March 2007.

[4] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, September 2000.

[5] An He, Joseph Gaeddert, Kyung Kyoon Bae, Timothy R. Newman, Jeffrey H. Reed, Lizdabel Morales, and Chang-Hyun Park. Develpment of a case-based reasoning cognitive engine for IEEE 802.22 WRAN applications. *Mobile Computing and Communications Review*, 13(2):37–48, April 2009.